Folsom Prerelease Release Notes

Flash Player 11.6 and AIR 3.6 Release Notes

Welcome to Adobe® Flash® Player 11.6 and Adobe AIR 3.6!

Last Updated:

This pre-release includes new features as well as enhancements and bug fixes related to security, stability, performance, and device compatibility for Flash Player 11.6 and AIR 3.6. This document may be updated periodically as more information becomes available.

NOTE:

- The ActiveX Flash Player in this release is not compatible with Windows® 8
  .Flash Player for Windows® 8 is available as part of the generally available Windows® 8 update

Release features

- **Full Screen Permission Dialog UI Improvement**
  Changed the location of the permission dialog to the middle of the screen. Also improved the user experience when going into full screen mode by adding the "Cancel" button.
- **Graphics Data Query**
  Developers can read the structure of the display object and vector data at runtime. Game developers can use this feature to create complex Sprite Sheets, or exporters for any file format (e.g. SVG) at runtime.
- **Multiple SWF Support**
  This feature provides support for packaging and loading multiple SWFs on iOS in AOT mode. With this feature, user can use multiple SWFs in an AIR iOS application using the Loader class.
- **Exclude devices from requestedDisplayResolution tag**
  This feature allows developers to explicitly disable the specified display resolution (high/standard) on one or more iOS devices using the excludeDevices attribute of the requestedDisplayResolution tag in the application descriptor.

Builds changes

Runtime Versions

Flash Player Desktop only: 11.6.602.105
AIR Runtime Desktop: 3.6.0.535
AIR Runtime Android: 3.6.0.535
AIR SDK: 3.6.0.535

Authoring

Authoring for Flash Player 11.6

To use the new Flash Player, you will need to target SWF version 19 by passing in an extra compiler argument to the Flex compiler: -swf-version=19. Directions are below. If you are using the Adobe Flex SDK:

- Download the new playerglobal.swc for Flash Player 11.6
- Download Flex 4.5.1 SDK (4.5.1.21328) from the Flex 4.5 SDK table.
- Install the build in your development environment
- In Flash Builder, create a new project: File -> New -> project.
- Open the project Properties panel (right-click and chose 'Properties'). Select Compiler from the list on the left.
- Use the 'Configure Flex SDKs' option in the upper right hand corner to point the project to Flex build 21328. Click ok.
- Configure your project to target SWF version 19
- Open the project Properties panel (right-click and chose 'Properties'). Select Compiler from the list on the left.
- Add to the 'Additional compiler arguments' input: -swf-version=19. This ensures the outputted SWF targets SWF version 19. If you compile on the command-line and not in Flash Builder, you need to add the same compiler argument.

Authoring for AIR 3.6 Update to the AIR 3.6 namespace

You must update your application descriptor file to the 3.6 namespace in order to access the new AIR 3.6 API's and behavior. If your application does not require the new AIR 3.6 API's and behavior, you are not required to update the namespace. However, we recommend all users start using the AIR 3.6 namespace even if you are not yet taking advantage of the new 3.6 capabilities. To update the namespace, change the xmlns attribute in your application descriptor to: <application xmlns="http://ns.adobe.com/air/application/3.6">

Feature Usage Guidelines

Multiple SWF support

This feature provides support for packaging and loading multiple SWFs on iOS in AOT mode. With this feature, user can use multiple SWFs in an AIR iOS application using the Loader class. There are a few limitations on iOS for using this feature:

1) The secondary SWF which is to be loaded by root SWF should have the same as the root SWF, otherwise loading will result in Error 3747: Multiple application domains are not supported on the operating system. Following is the correct way to load a secondary SWF:

Example.as

```
var aLoader:Loader = new Loader();
var url:URLRequest = new URLRequest("swfs/SecondarySwf.swf");
var loaderContext:LoaderContext = new LoaderContext(false, ApplicationDomain.currentDomain, null);
aLoader.load(url, loaderContext); // load the SWF file
```

2) Methods unload() and loadBytes()of the loader class will not work on iOS.
3) Number of SWFs that can be packaged in an application will depend on the capability of the machine, because while packaging the IPA, machine can go out of memory and therefore, packaging will fail with out of memory error.

Graphics Data Query

This feature allows you to query any DisplayObject and get a representation of it through GraphicsData objects. This is very useful to serialize/deserialize a DisplayObject, create custom exporters (spritesheets, SVG, etc.).

For more details about this feature: http://www.bytearray.org/?p=4893

Exclude devices from requestedDisplayResolution tag

A new attribute ‘excludeDevices’ has been added to the <requestedDisplayResolution> tag in the application descriptor. This attribute will allow developers to explicitly disable the specified display resolution on one or more iOS devices. To use this feature, application descriptor namespace 3.6 or greater would be required. This feature will not be supported on AIR simulator. A developer could exclude:

- A particular device by mentioning its exact model name. Following example disables retina display only on iPad with device model as iPad3,1.

  ```
  <requestedDisplayResolution excludeDevices=”iPad3,1”>high</requestedDisplayResolution>
  ```

- Multiple devices by providing a space separated list of exact model names. Following example disables retina display only on iPads with device model name as iPad3,1 or iPad4,1.

  ```
  <requestedDisplayResolution excludeDevices=”iPad3,1 iPad4,1”>high</requestedDisplayResolution>
  ```

- All Variations of a particular model. Following example disables retina display on all variations of 'iPad3' like iPad3,1 iPad3,2.

```
<requestedDisplayResolution excludeDevices=”iPad3”>high</requestedDisplayResolution>
```

- A family of devices. Following example disables retina display on all iPhones (irrespective of the model)

```
<requestedDisplayResolution excludeDevices=”iPhone”>high</requestedDisplayResolution>
```

Similarly, retina mode can be enabled for particular devices by excluding from this list when requestedDisplayResolution is specified as standard in the application descriptor. The following example enables retina display only on iPhone (all models) while apps continue to run using the standard display resolution on other devices.

```
<requestedDisplayResolution excludeDevices=”iPhone”>standard</requestedDisplayResolution>
```

Note. Device model name can be fetched using the flash.system.Capabilities.os property. The following table lists the device model names for commonly used iOS devices:

Device	Model Name
iPod Touch Fourth Generation	iPod4,1
iPod Touch Fifth Generation	iPod5,1
iPhone 3GS	iPhone2,1
iPhone 4	iPhone3,1
iPhone 4 CDMA	iPhone3,2
iPhone 4S	iPhone4,1
iPhone 5	iPhone5,1
iPad	iPad1,1
iPad 2	iPad2,1
iPad 2 (GSM)	iPad2,2
iPad with Retina display(A5) (CDMA)	iPad2,3
iPad with Retina display(A5) (CDMAS)	iPad2,4
iPad Mini (Wifi)	iPad2,5
iPad with Retina display(A5) (Wifi)	iPad3,1
iPad with Retina display(A5) (CDMA)	iPad3,2
iPad with Retina display(A5) GSM	iPad3,3
iPad with Retina display(A6X) (Wifi)	iPad3,4

Preview of "extended profile" added to Stage3D

Please note that this feature is included in the Beta releases for everyone to try out the feature and provide feedback. This feature will be turned off before we ship Flash Player 11.6 and AIR 3.6.
Please, remember that this feature is in beta, you may encounter some instability for this feature during the few beta releases.
The new additional features and functionality to Stage3D provide advance capability on Desktop to enable higher end hardware to do high end rendering.

To enable extended profile, use the Context3D.BASELINE_EXTENDED constant for the Stage3D.requestContext3D call.

Please note that the extended mode is currently only available on desktop hardware and there is no software fallback for this mode.
To use Stage3D Extended mode please publish the swf as version 19.

New texture format, half float, rgba, 64 bit, 16bit per channel.

Use the constant Context3DTextureFormat.RGBA_HALF_FLOAT for creating such textures.

This is in early stages and more additional formats like full float and dual channel textures as well as packed and srgb will be supported in the next beta.

You can render to floating point textures and use them with rectangle textures for deferred rendering algorithms.

Separate sampler state

This functionality allows the setting of texture sampling flags such as linear/nearest neighbor filtering and clamp/repeat from outside of the fragment program. This increases Stage3D flexibility to fit into existing frameworks and workflows.

This functionality is available in all profiles.

Use the function
Context3D.setSamplerStateAt after setting an AGAL program using Context3D.setProgram to manually override the filtering and repeat properties of a sampler.
Setting a proram through Context3D.setProgram will restore samplers to what is specified in the program.
If that behavior is not desired, set the "ignoresampler" property in AGAL.

Multiple render targets.

The setRenderToTexture function now accepts an index. Up to 4 targets can be simultaneously rendered to.
All targets must be of the same bit depth. Please note that some error checking for this functionality has not been implemented yet in this Beta.

Rectangle textures

Allows textures to be non-square, non-power of two sizes. This allows the texture to be the exact size of the back buffer which is essential to deferred rendering technique.

Context3D.createRectangleTexture

Rectangle textures do not need to be powers of two in size but do not support repeat mode or mip mapping.
This functionality is currently available only in extended profile but we are currently considering supporting this in other other profiles.

AGAL version 2. Please use a new AGALMiniAssembler to emit the version 2 token in the header.

With AGAL version 2 you can use the following new features:
You can download the AGAL version 2 at http://bit.ly/UEjqbz

Fragment Program Depth Output:

Instead of writing to the standard fragment program 'Output Color', this will enable 'Output Depth'. When 'Output Depth' is written to from the fragment program it will be used instead of the depth output from the vertex program. This is useful for 2D games to achieve isomorphic effect, where each pixel of an image can have a different depth value.

Write the depth output register od0.x, a scalar value in the 0..1 range that is used for depth testing and writing
Write to multiple render targets oc0..oc3 for color values.

Larger register pool for Program3D

To allow more complex rendering we have increased the register limits for Program3D.

Type	Baseline	Extended
Vertex streams	8	8
Vertex constants	128	250
Fragment constant	28	64
Temporaries	8	26
Varying	8	10
Sampler	8	16
Output colors	1	4
Depth output	0	1

Token count	200	1024

New instructions for conditional forward branches:

Enable if/else branches in AGAL based on a scalar comparison. The comparison can be any of these ==, !=, >, >=. Based on the result of the comparison you can skip ahead to a future part of the program. This enables more efficient program execution which will help mitigate the use of heavier programs in terms of performance.

Mnemonic	Value	Description
ife	0x1c	Jump if source1 is equal to source2
ine	0x1d	Jump if source1 is not equal to source2
ifg	0x1e	Jump if source1 is greater or equal than source2
ilt	0x1f	Jump if source1 is less than source2
els	0x20	Else
eif	0x21	Close an if or else block

New instructions for low precision partial derivative in fragment programs:

Calculates the rate of change between the color of two pixels in the x or y direction. This allows for custom antialiasing solutions as well as edge effects.

Mnemonic	Value	Description
ddx	0x1a	Load partial derivative in X of source1 into destination
ddy	0x1b	Load partial derivative in Y of source1 into destination

Those opcodes are only available in the fragment program.

New texture instruction with computed LOD:

With the use of conditional jumps in the fragment shader traditional texture fetch commands will not work because GPU drivers operate on pixel batches in lock step and a dynamic level of detail cannot be determined for each batch while in a conditional (because some pixel batches may enter the conditional while others may not). Because of this the mip level (level of detail) must be supplied at compile time. This new opcode allows the specification of an explicit LOD.

Mnemonic	Value	Description
ted	0x26	Texture fetch with explicit LOD in w component of the texture coordinate

This instruction is required for texture fetches inside a conditional block.

Reporting issues

Found a bug? Submit a bug report at http://bugbase.adobe.com

Note: Flash Player and AIR may leverage your sound and graphics hardware for optimal performance and power usage. Some issues may be specific to the particular combination of hardware and drivers on your system. When reporting an issue involving graphics, video or stability, it is essential to note your graphics and sound hardware, their corresponding driver versions, your operating system and browser version (when using Flash Player).

For details on how to collect this information, see: Instructions for Reporting Video Playback Issues.

System Requirements

For current Flash Player system requirements visit http://www.adobe.com/products/flashplayer/systemreqs/
For current AIR system requirements, visit http://www.adobe.com/products/air/systemreqs/

Flash Player 11.6 has the following minimum system requirements:

	Windows	Macintosh
Processor	2.33 Ghz or faster x86-compatible processor, or Intel® Atom™ 1.6GHz or faster processor for netbook class devices	Intel® Core™ Duo 1.83GHz or faster processor
Operating System	Microsoft® Windows® XP (32-bit), Windows Server® 2003 (32-bit), Windows Server 2008 (32-bit), Windows Vista® (32-bit), Windows 7 (32-bit and 64-bit)	Mac OS® X 10.6, Mac OS X 10.7
Browser	Internet Explorer 7.0 and above, Mozilla Firefox 4.0 and above, Google Chrome, Safari 5.0 and above, Opera 11	Safari 5.0 and above, Mozilla Firefox 4.0 and above, Google Chrome, Opera 11
Memory	128MB of RAM (1GB RAM recommended for netbook class devices), 128MB of graphics memory	256MB of RAM, 128MB of graphics memory

AIR 3.6 has the following minimum system requirements:

	Windows	Macintosh	Android	iOS
Processor / Device Hardware	2.33GHz or faster x86-compatible processor or Intel Atom™ 1.6GHz or faster processor for netbook class devices	Intel® Core™ Duo 1.83GHz or faster processor	ARMv7 processor with Vector FPU, Minimum 550MHz, ES2.0, H.264 & AAC H/W Decoders	iPod touch 4, iPod touch 5, iPhone 3GS, iPhone 4/4S, iPad, iPad 2, iPhone 5, The New iPad, iPad mini
Operating System	Microsoft® Windows® XP, Windows Server® 2003, Windows Server® 2008, Windows Vista® Home Premium, Business, Ultimate, or Enterprise (including 64-bit editions) with Service Pack 2, or Windows 7	Mac OS® X 10.6 and 10.7	Android 2.2, 2.3, 3.0, 3.1, 3.2 & 4.0	iOS 5.0 and higher
RAM	512MB of RAM (1GB recommended)	512MB of RAM (1GB recommended)	256MB RAM	-